**In the claims:**

All of the claims standing for examination are reproduced below with appropriate status indication.

1. (Currently amended) A software application for enabling automated notification of applied structural changes to electronic information pages hosted on a data packet network comprising:

a developer-interface module for enabling developers to build and modify network navigation and interaction templates using functional logic blocks, for automatically navigating to and interacting with interactive electronic information pages on the data packet network;

a navigation system-interface module for integrating the software application to a proxy-navigation system for periodic execution of the templates;

a change-notification module for indicating a point in process where a navigation and interaction routine has failed and for creating a data file containing parameters associated with the failed routine; and

a database interface module for interfacing the software application to a data repository for storing the data file, wherein the software application periodically submits test navigation and interaction routines to the navigation system for execution by virtue of the interface with the navigation system, and upon failure of a test routine, creates the data file, the data file comprising a point-of-failure indication within the failed routine, an identifier of the associated electronic information page subjected to the navigation routine, and stores the data file in the data repository sending notification of the action to the developer.

2. (Original) The software application of claim 1, wherein the data-packet-network is the Internet network.

3. (Original) The software tool of claim 2, wherein the functional logic blocks include site-logic blocks, automated site-login blocks, and automated site-registration blocks.

4. (Original) The software application of claim 3, wherein the software application is an Internet-based application executing and running on an Internet server.

5. (Original) The software application of claim 4, wherein the software application is accessible through a network-browser application.

6. (Original) The software application of claim 5, wherein the navigation templates are test routines executed for the purpose of determining success or failure of the routine.

7. (Original) The software application of claim 6, wherein the navigation templates are executable instruction orders containing logic blocks.

8. (Original) The software application of claim 7, wherein the functional logic blocks are modular and self-installable within the navigation templates.

9. (Original) The software application of claim 8, wherein the data files are human readable and are accessed by developers for the purpose of affecting updating of the navigation templates.

10. (Original) The software application of claim 9, wherein the developers access the application through individual computerized workstations.

11. (Original) The software application of claim 10, wherein the error notification and data file creation processes are also performed in the event of failure of a client's personalized navigation template.

12. (Previously presented) A change-notification system for detecting structural changes applied to electronic information pages hosted on a data-packet-network comprising:

a software application installed on a network-connected processor, the software application enabling developers to construct and cause execution of navigation and interaction templates and enabling failed instances of navigation and interaction executed on the network to be reported;

a server system connected to the network, the server system hosting a proxy-navigation software application for executing the templates, the navigation software accessible through the software application;

a data repository accessible to the server system and to the software application, the data repository storing information about clients and result information about the failed navigation and interaction routines, the result information supplied by the software application; and

a plurality of network-connected nodes having network access to the software application and to the data repository, wherein access of the software application is practiced by developers operating the network-connected nodes for the purpose of building and causing execution of the navigation and interaction templates, the templates used to test the current structural states of electronic information pages hosted on the network, and wherein the software application notifies of failure instances of the executed routines, the failure instances logged in the database.

13. (Original) The change-notification system of claim 12, wherein the data-packet-network is the Internet network.

14. (Original) The change-notification system of claim 13, wherein the network-connected processor hosting the software application is an Internet-connected server.

15. (Original) The change-notification system of claim 14, wherein the server system hosting the proxy navigation software also hosts the software application.

16. (Original) The change-notification system of claim 15, for in the server system contains a single server hosting both the proxy navigation software and the software application.

17. (Original) The change-notification system of claim 16, wherein the software application and the proxy navigation software are integrated as a single application enabling both functions of navigating according to navigation templates and notifying and recording failed instances of navigation.

18. (Currently amended) A method for receiving notification of random structural changes applied to electronic information pages accessed by a proxy network navigation and interaction system and effecting updates to navigation templates based on the change information comprising steps of:

(a) establishing notification of a failed navigation and interaction routine executed for the purpose of navigating to and interacting with an electronic information page on a data-packet-network;

(b) recording an instance of the failed routine including parameters associated with the cause of failure;

(c) accessing the recorded instance of the failed routine for review purposes;

(d) navigating to the electronic information page identified in the recorded instance on the data-packet-network;

(e) accessing source information associated with electronic information page identified in the recorded instance;

(f) creating Withdrawn logic according to the source information and according to information contained in the recorded instance; and

(g) installing the Withdrawn logic into existing navigation templates that depend on the updated information for successful function.

19. (Original) The method of claim 18, wherein the data-packet-network is an Internet network and electronic information page is a web page hosted on the network.

20. (Original) The method of claim 19 wherein in step (a), wherein the navigation routine is performed according to a test navigation template created for the purpose.

21. (Original) The method of claim 19 wherein in step (a), wherein the navigation routine is performed according to a client navigation template executed to perform services for the client.

22. (Original) The method of claim 19 wherein in step (b), the recorded instance of a failed routine is created in the form of a data file and stored in a data repository accessible through the network.

23. (Original) The method of claim 22 wherein in step (c), the recorded instance of the failed navigation routine is accessed by a human software developer.

24. (Original) The method of claim 23 wherein in step (d), navigation is performed by the developer utilizing an instance of browser software installed on a computerized workstation.

25. (Original) The method of claim 24 wherein in step (f), the Withdrawn logic is credited in the form of a modular logic block installable to navigation templates.

26. (Original) The method of claim 25 wherein in step (g), the Withdrawn logic block self-installs to a depended navigation template.

27. (Original) The method of claim 18 wherein a step is added between steps (f) and (g) for testing the Withdrawn logic before implementation.

28. (Original) The method of claim 26 wherein in step (g) more than one Withdrawn logic block is created for a single navigation template.

29-86. (Canceled)